Attorney's Docket No.: 07844-315001
Client's Ref. No.: P289

OFFICIAL COMMUNICATION FACSIMILE:

OFFICIAL FAX NO: (571) 273-8300

Number of pages including this page 7

Applicant : Gordon B. Dow
Serial No. : 09/293,737
Filed : April 16, 1999

Art Unit : 2194
Examiner : Charles E. Anya

Title : DYNAMIC DEPENDENCY GRAPH IN MVC PARADIGM

MAIL STOP AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Attached to this facsimile communication cover sheet is NOTICE OF APPEAL; PRE-APPEAL BRIEF REQUEST FOR REVIEW, faxed this 9th day of November, 2005, to the United States Patent and Trademark Office.

Respectfully submitted,

Date: November 9, 2005

Daniel J. Burns
Reg. No. 50,222

Fish & Richardson P.C.
500 Arguello Street, Suite 500
Redwood City, California 94063
Telephone: (650) 839-5070
Fax: (650) 839-5071

50311334.doc

Attorney's Docket No.: 07844-315001 / P289

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant : Gordon B. Dow
Serial No. : 09/293,737
Filed : April 16, 1999
Title : DYNAMIC DEPENDENCY GRAPH IN MVC PARADIGM

Art Unit : 2194
Examiner : Charles E. Anya

Mail Stop AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

PRE-APPEAL BRIEF REQUEST FOR REVIEW

This brief is in response to legal and factual deficiencies in the non-final Office Action mailed August 9, 2005.

I. The Cited Art does not Teach or Suggest Severing Dependencies

Objects that use the value of other objects to compute their own value are dependent on those other objects. *See* Specification, pp. 5-6. With reference to FIG. 1B below, object C (102c) depends directly on object A (102a), whereas object D (102d) depends directly on object C and indirectly on object A. Dependencies between objects are maintained explicitly in one or more dependents list 104 data structures whose linkages create a graph of dependency relationships.

CERTIFICATE OF TRANSMISSION BY FACSIMILE

I hereby certify that this correspondence is being transmitted by facsimile to the Patent and Trademark Office on the date indicated below.

November 9, 2005
Date of Transmission

Signature

Sarah E. Hoke
Typed or Printed Name of Person Signing Certificate

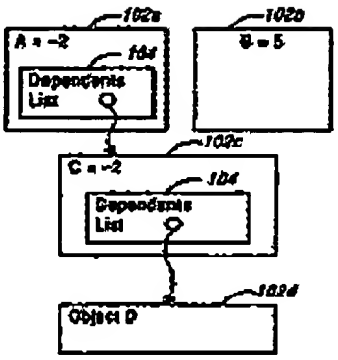


FIG. 1B

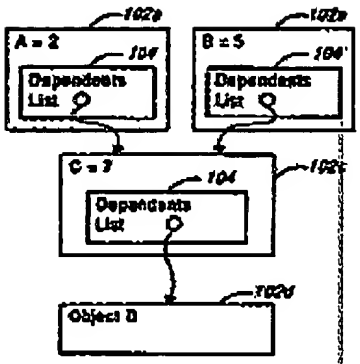


FIG. 1C

In this illustration, object C (102c) has a value defined as a function:

$$C = \text{if } (A < 0) \text{ then } A \text{ else } (A + B).$$

When the value of object A is less than zero, object C depends only on object A, as shown in FIG. 1B. This dependency is represented explicitly in object A's dependents list. When the value of object A is equal to or greater than zero, object C depends on both objects A and B (102b) as shown in FIG. 1C above.

When the value of an object changes, all objects that depend directly or indirectly on the changed object are marked as invalid and their path in the dependency graph to the changed object is severed. With reference to FIG. 1B, if the value of object A changes, reference to object C in object A's dependents list is removed and reference to object D in object C's dependents list is removed.

This feature is recited in independent claims 1 and 28 in this way: "when the value of object B changes, invalidating the dependents of object B and all of their further dependents, including severing dependencies among the dependents of object B and all of their further dependents."

This feature is recited in independent claims 11 and 31 in this way: "severing dependencies from the changed object and all of its direct and indirect dependent objects."

This feature is recited in independent claims 15 and 39 in this way: "for each dependent object on the dependency graph, marking the dependent object as dirty and detaching the dependent object from the dependency graph."

This feature is recited in independent claim 35 in this way: "when the value of object B changes, the dependents of object B and all of their further dependents are invalidated, and the dependencies among the dependents of object B and all of their further dependents are severed."

The feature of severing dependencies is clearly called for in the above claims. However, the cited references plainly do not teach this feature. The Examiner conceded that Razdow (U.S. Pat. No. 5,469,538) and Conway (U.S. Pat. No. 6,272,672) are silent with respect to severing dependencies. The relied upon portion of Wu (U.S. Pat. No. 5,404,428) teaches that when a view model attribute is modified by an application program, derived items dependent upon that attribute are invalidated but not severed from the acyclic graph. *See* Wu, col. 9, lines 1-47. Finally, the cited portion of Picott (U.S. Pat. No. 5,929,864) illustrates pseudo code only for marking dependents as dirty. *See* Picott, col. 8, line 29.

II. The Cited Art does not Teach or Suggest Calculating a Dependency Among Objects at the Time Objects Calculate Their Values

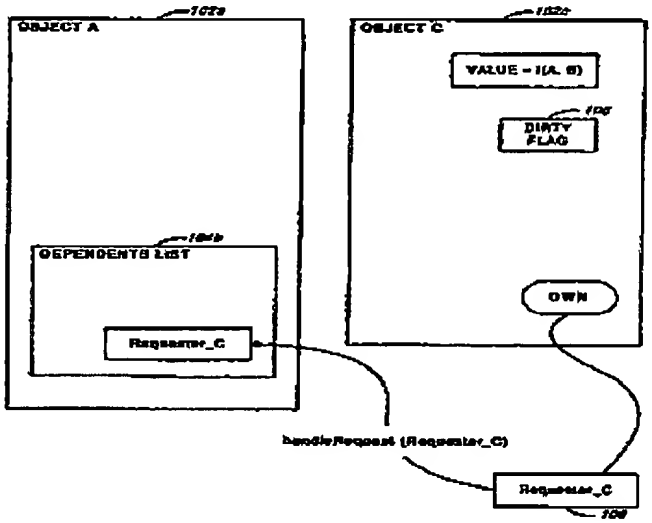
When an object needs to recompute its value, dependencies are established with all of the other objects it depends on. *See* Specification, p. 5. With reference again to FIGS. 1B and 1C, when object A's value changes from -2 to +2, objects C and D's dependencies on object A would be severed as discussed above. When object C calculates its value, new dependencies between object C and objects A and B are established as shown in FIG. 1C.

This feature is recited in independent claims 19, 33 and 40 in this way: "calculating the dependency among objects in the set dynamically *at the time objects calculate their values*."

Conway fails to teach or suggest this feature, as does Bardasz (U.S. Pat. No. 5,689,711). In contrast to calculating dependencies, the cited portion of Conway merely describes message sending between components during recomputation of outputs. *See* Conway, FIG. 6 and col. 21, lines 1-46.

According to Bardasz, "[d]ata objects are components in the graph that contain values." Bardasz, col. 6, lines 46-47. "Operators are components in the graph that take action on at least one data object and are functions or evaluable expressions." Bardasz, col. 6, lines 50-53. The relied upon text in Bardasz discloses determining dependencies of operators on data objects. *See* Bardasz, col. 20, lines 18-25. Thus, the cited text fails to teach or suggest calculating dependencies among data objects. Furthermore, the dependencies between operators and data objects already exist in the dependency graph of Bardasz *before* operator components are evaluated.

III. The Cited Art does not Teach or Suggest Identifying the Objects upon which a Given Object depends as those Objects into which the Given Object Passed Itself as a Requester during Execution of a Compute Method of the Given Object



This feature of Applicant's claims is specific to an object oriented programming paradigm. A requester is a class used during calculation of object values. *See* Specification, pp. 8 and 17, and FIG. 1D above. For example, a dependency is established between object A and object C when an object C's value is calculated. During the calculation, object C passes itself in

a requester 108 to object A, which object C's calculation depends on. As a result, a reference to object C is added to Object A's dependents list 104.

This feature is recited in independent claims 7, 29 and 36 in this way: "identifying the objects upon which a given object depends as those objects *into which the given object passed itself as a requester* during execution of a compute method of the given object."

The cited portions of Razdow describe representing numeric equations as dependency graphs, but do not discuss this feature, much less object oriented programming. *See* Razdow, col. 8, lines 21-4; col. 11, lines 39-61. Likewise, the relied upon portion of Picott merely describes message sending between nodes in a directed acyclic graph *See* Picott, col. 7, lines 28-35. Finally, the Examiner depends on Conway but the cited text only discuses message sending between components, not creating dependencies between components by passing objects. *See* Conway, col. 21, lines 25-55 and FIG. 6.

II. Conclusion and Relief

Applicant already filed an Appeal Brief on April 18, 2005. The Examiner's response in the above-cited office action introduces new references but commits the same errors. The rejections of record are clearly improper and without basis.

Please apply any charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: 11-9-2005

Daniel J. Burns
Reg. No. 50,222

Fish & Richardson P.C.
500 Arguello Street, Suite 500
Redwood City, California 94063
Telephone: (650) 839-5070
Facsimile: (650) 839-5071
50311233.doc